instruction queue 240 and the SROM 250 are executed. In this fashion, the program flash 260 is bypassed.

In one embodiment, a command may be placed in the instruction queue 240 by the test controller 120, which may cause the microprocessor 210 to go into a supervisory state. In this state, the microprocessor 210 may take control of the multiplexer 270. In this fashion, commands from the supervisory ROM 250 are run in the microprocessor 210. The microprocessor 210 may also cause selected logic (e.g., registers 230), which was not accessible, to become accessible. After the commands have finished, the microprocessor returns control of the mulitplexer to the test controller and once again commands are executed from the instruction queue 240.

Thus, embodiments may execute a series of instructions by the test controller 120 filling the instruction queue 240 with the instructions. When executing commands from the supervisory ROM the circuit 150 may be tested at normal clock speed. Embodiments allow running complex testing algorithms at high speed by switching between instructions that the test controller 120 put in the instruction queue 240 and pre-determined sets of instructions in the supervisory ROM 250. Thus, an instruction in the instruction queue 240 may function as a subroutine call and the supervisory ROM 250 may store a number of subroutines.

The program flash 260 may contain instructions that may be executed in the microprocessor 210 during normal mode (e.g., not test mode). Embodiments bypass the program flash 260 when in test mode.

The test controller 120 may transfer data between itself and the peripheral registers 230 and the Static RAM (SRAM) 220 via the bus. Thus, the contents of the peripheral registers 230 and SRAM 220 may be read and written from the external controller 100, as explained in the description of Figure 1.

The test controller 120 is able to transfer commands to the instruction queue 240 via the bus 201. As explained herein, the test controller 120 may receive commands via the test interface 110 from the external controller 100 and translate these commands, if necessary.

In the present embodiment, the program flash 260 may only receive data from the microprocessor 210. The instruction queue 240, Supervisory ROM (SROM) 250, and the Program Flash 260 are addressed by the microprocessor 210.

It will be understood that the various registers and memories may be implemented with other hardware. For example, the SROM 250 may be implemented with RAM or otherwise. The same applies to the program flash 260 and other elements.

The peripheral registers 230 may be, in general, those components which constitute parts of a microcontroller. For example, these registers may comprise programmable systems on a chip (PSoCs (both analog and digital)), input/output pins, and various fixed function blocks.

Thus, embodiments of the present invention provide for an efficient mechanism for feeding instructions to the microprocessor 210, single stepping instructions, programming the analog and digital PSoCs, routing the digital and analog PSoCs, routing input and output signals to I/O pins, and executing programs without requiring flash memory 260.

Figure 3 illustrates an embodiment of the present invention in which the test interface 110 is shared with a crystal oscillator 310. Figure 3 also shows an additional test port 320 which may be used instead of the test controller 120. A multiplexer 370 is used to select whether the test controller 120 or additional test port 320 has access to the test interface 110.

APPLYING INSTRUCTIONS TO A MICROPROCESSOR WHILE IN TEST MODE

Embodiments allow instructions to be fed to a microprocessor 210 from both an external source and from an on-chip source. For example, arbitrary instructions may be entered via the test interface 110. Furthermore, the supervisory ROM 250 may store pre-determined sub-routines. Various embodiments may single step through a series of instructions loaded into the instruction queue 240, may execute instructions from the Supervisory ROM 250, and may alternate between the two sources, allowing for great flexibility.

Figure 4 illustrates an embodiment of a process 600 in which multiple instructions are single stepped though the instruction queue 240, along with the option of running routines from the SROM 250. In step 610, a test mode is entered